Parallel Climate Data Assimilation PSAS Package

Achieves 18 GFLOPS on 512-node Intel Paragon

Hong Q. Ding, Clara Chan, Donald B. Gennery, and Robert D. Ferraro
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109

ABSTRACT. We have designed and implemented a set of highly efficient and highly scalable algorithms for an unstructured computational package, the **PSAS** data assimilation package, as demonstrated by detailed performance analysis of systematic runs onupto512node Intel Paragon. The equation solver achieves a sustained 18 Gflops performance. As the results, we achieved an unprecedented 100 -fold solution time reduction on the Intel Paragon parallel platform over Cray C90. This not only meets and exceeds the DAO time requirements, but also significantly enlarge the window of exploration in climate data assimilations.

1.0 Overview

An important aspect in short-term numerical weather prediction and long-term climate modeling is incorporating the observational data into the simulation systems, Since observational data come with various uncertainties and errors, the data are incorporated in a statistical sense, i.e., they are filtered through a Kalman filter. Because the observational points are irregularly distributed on the surface of the earth (both latitude, longitude and elevation), and they change from time to time, the filtered observations must to be interpolated to regular grids on which model systems are based.

The Physical-space Statistical Analysis System (PSAS) developed at the Data Assimilation Office (DAO) at Goddard Space Flight Center[1] is an advanced system which provides a general framework to perform the above data assimilation tasks. This software system will play a central role in Mission to Planet Earth enterprise and is designated to replace the existing operational system at the DAO by 1998. Currently, a brute-force application of PSAS for a complete analysis requires about 4-7hrs on Cray C90. This falls far short of the DAO requirement of 120 re-analyses per day in real time.

Recently, we have implemented all major parts of the PSAS package on the Intel Paragon. We designed and incorporated several new algorithms which are highly efficient and scale well to very large numbers (thousand) of processors. For example, a time-critical part of the package is the sparse linear equation solver, which achieves a sustained speed of 18.3 GFLOPS on a 5 12-node Paragon (see Sec.5). This represents 36% of the theoretical total peak speed of the 512-node Paragon.

As a result, the parallel PSAS package solves an 80,000 observation problem in just 176 sees (wall clock time, including about 30 sees spent on reading/writing data from/to disk) on the 512-node Paragon. In contrast, the same problem takes an estimated 4-5hr CPU time on the Cray C90 (the first half took 9120 sees on C90 vs. 76 sees on Paragon, and the second half is estimated to take about the same time on C90). This represents an unprecedented 100-fold reduction in solution time. This parallel PSAS not only meets and substantially exceeds the time requirements; in fact it will change the DAO operations significantly: many previously unexplored problems due to hugh computational requirements now can be solved in a timely manner.

2.0 Basic formulation

Inmost computer models, theweather system on the surface of the Earth is represented by a regular 20x2.50 grids, using the familiar latitudes and longitudes. The entire atmosphere is covered. The grids along the vertical dimension is marked by pressure levels, usually 1000mbar to 20 mbar (elevation is converted to pressure using thermostatic equations). These grids remains fixed during the simulation time period (1 month, 10 year, etc.). The number of model variables N_a (all components at all pressure levels) is around 10^6, and will increase as we increase pressure levels and/or refine horizontal grids.

The observation data, many of them from satellite observations, are not uniformly distributed on earth; some parts of the world have few observation data. Further complicating the situation is that both the number of observation data and their location change from time to time, making it impossible for systematic comparison over time. At present, the number of observations No is around 100,000, and will grow with time. The PSAS algorithm transforms these observations from diverse sources at arbitrary space locations to physically-consistent data set on the above mentioned regular model grids, therefore propagating information from observed regions to unobserved regions,

Given a model forecast w_f (how the forecast is calculated and evolved in time are independent of the data assimilation process), and the actual observations W. which comes with uncertainties as reflected in the error covariance matrix $\boldsymbol{R}$, the statistical process is to obtain an optimal estimate w_a of the state of the weather system through a Kalman filter,

$$w_a = w_f + K(w_o - Hw_f)$$

where H is a interpolation operator interpolating forecast from grid locations to observation locations, and $K = P^f H^T (HP^f H^T + R)^{-1}$ is the Kalman gain matrix, Here P contains physical correlations calculated using a large number of existing routines. Clearly, calculating the Kalman gain matrix K by inverting a $N_a \times N_a$ matrix is a very hard problem. Instead, the PSAS algorithm solves the following equation for the vector x

$$(HP^f H^T + R)x = w_o - Hw_f$$

and then fold back the solution from observation locations to the grids location to obtain the final optimal analysis state

$$w_a = w_f + P^f H^T x$$

In other words, the net effect of incorporating the observation data through the Kalman filter is the increment $\Delta w = P^f H^T x$ to the forecast. In the next two sections, these two parts of PSAS are discussed in tail.

Since both parts relate to observations which are irregular in space and change in different runs, the PSAS problem is an unstructured problem whose parallel implementation are generally more complicated than regular problems. Also, the physics, i.e., the large number of correlations between different components, such

as winds, water vapor mixing ratios, are rather complex (about 7500 lines Fortran codes spread in 40 subroutines).

We take a disciplined modular approach (maybe object-oriented approach is better?) in which all high level data organizations are done using structures in C while leaving lower level of physics details in Fortran; interfaces to low level are rewritten to restrict accesses through only the matrix block assemble and one or two C structures. All new codes, including the preconditioned CG which is essentially a skeleton code calling underlying Fortran routines for numerical calculations, are written in C, and use the highly efficient BLAS routine whenever possible. This approach makes modification of physical problem much easier and at same-time maintains the efficiency of Fortran for numerical computations.

3.0 Solving the Correlation Equation

The critical part of the PSAS is the solution of a large linear system of equations, the correlation equation, $Mx=b$, with 10^5 unknowns. The challenges of the problem lie in the size of the matrix involved, The symmetric correlation matrix $M = HP^fH^T+R$ has a size of $10^5 x 10^5$ with 26% nonzero, due to the cutoff approximation of the correlations at 6000km on the surface. To store the entire matrix would require 10 GB (in single precision, or 20 GB in double precision) computer memory, exceeding the capacities of any existing sequential computer. This difficulty is resolved in the Cray C90 codes by re-computing the matrix on the fly, at considerable expense of CPU time.

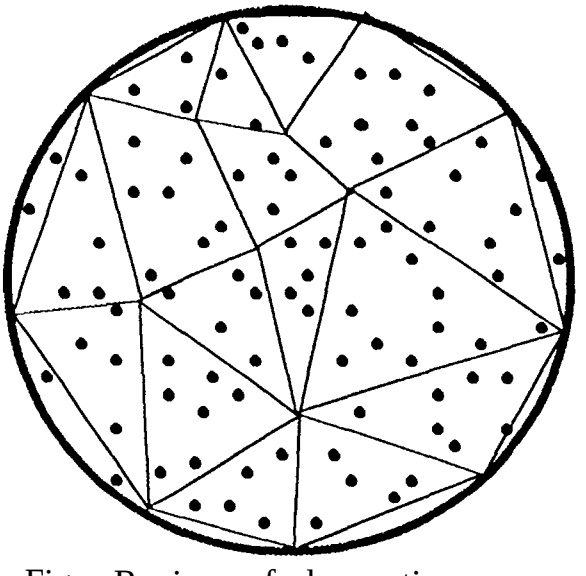

Fig. Regions of observations,

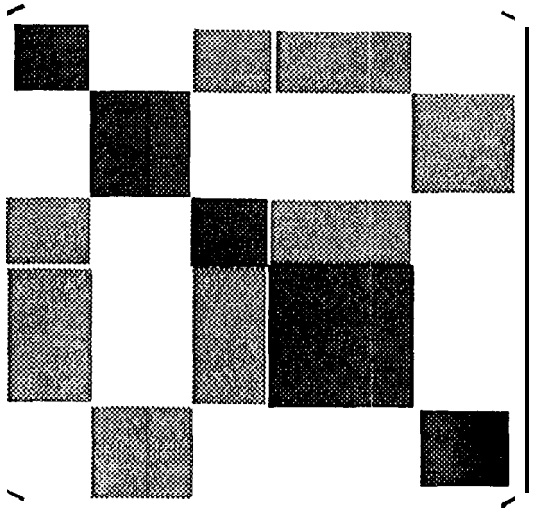

Fig.2 Block structure of the correlation matrix. Only nonzero blocks are shown.

The memory-bound problem fits well to distributed-memory parallel architecture, which could have much large total residence memory, The "sparse" correlation matrix, however, does not fit to conventional sparse matrix techniques (for matrices of nonzeros typically far lower than 26%, maybe around 2% or less). This difficulty is resolved by imposing a structure to the sparse matrix: observations are divided into regions with equal numbers using a concurrent partitioned we have previously developed[2], and the correlation cutoff is enforced at region level (see Figure 1). This approximation introduces a small error (our rigorous comparison

indicates a 1 -2Y0 rms error in solution), but increase the calculation speed dramatically. The imposed structure is a block structure with 74% of matrix blocks are identically zero (see Figure 2). Now the matrix-vector multiplication is carried out at BLAS 2 level speed, instead of the scalar speed due to the indirect indexing in conventional sparse matrix techniques. As the result, our solver runs at 77 MFLOPS on l-node Paragon, in contrast to about 10MFLOPS oft ypical scalar speed.

After the observations are concurrently partitioned into regions and properly distributed among the processors in a balanced way, we come to the distribution of the huge sparse matrix. A large number of nonzero matrix blocks of this irregular problem must be distributed among the processors in a load-balanced way (e.g., there are 34907 matrix blocks in the 512-node case). This is an optimization (linear programming) problem on 34907 variables with various constraints. We designed a fast iterative algorithm which finds a near-optimal distribution in just a few seconds.

Once the correlation matrix blocks are generated according to the distribution, the correlation equation is solved by a conjugate-gradient iterative solver, of which the key part is the global matrix-vector multiplication[3]. Given the imposed matrix block structure, the multiplication proceeds similar to the parallel block approach for dense matrix-vector multiplication. However, there are two important differences, First, we only store the upper-right matrix blocks due to symmetry; this allows each non-diagonal matrix block to be used twice in each matrix-vector multiplication, and therefore increases computation/communication ratio. Second, the communication here is irregular due to the absence of 74% of matrix blocks (which would exist in dense matrix case); and storing only upper-right half matrix adds more irregularity to the communication pattern. When everything is properly implemented, this new algorithm for the "not-so-sparse." sparse matrix-vector multiplication is highly efficient and scales well to large number of processors, as indicated by the performance numbers shown above.

The observations partitioned, the matrix blocks distributor and the PCG solver parts have been integrated and fully debugged with a system of monitoring and debugging, in which by changing a few runtirne parameters such as debug-flag, print-proc, print-flag, etc., details of a portion of the codes on a given processor, or a statistical summery of some important things over a portion or all processors will be automatically printed out to screen for monitoring or debugging purposes.

4.0 Folding back

Folding back solution to all pressure levels, i.e., calculating Aw = $P^f H^T x$, represents a very significant computational task, mainly in assembling the N_a x No matrix $P^f H^T$. At the 2°x2.50 horizontal resolution, there is 13104 grids at each pressure level. The smallest model system contains 14 pressure levels for four upper-air components and 3 sea-level components; they add up to N_a = (3+4x14) x13104 = 773,136 model variables. The operational system will have 1&22 pressure levels, which bring N_a to 10^6 or more. As mentioned before, the number of observations N_o is around 105 or more for operationals ystern. The matrix $P^f H^T$ has similar sparsity pattern as the correlation equation matrix, except it is not symmetric any more. The folding-back process is dominated by assembling the huge matrix $P^f H^T$. It is fortunate that the matrix-vector multiplication is carried out only once, so that the entire matrix does not need to be stored at same time; they are computed as needed on the fly.

The parallel implementation takes the advantage that the observations have been grouped into regions and distributed among the processors in a balanced way during the partitioning process. The model variables are partitioned among the processors in the beginning of the fold-back process. Once again, they are grouped into regions to efficiently implement the correlation cutoff at 6000km, which determines the sparsity of the folding matrix.

The partition of forecast model variables are based on 128 grid regions, which is different from those observation regions in shape, size and location. The 13104 grids on the 2°x2.50 mesh are grouped in 128 static rectangle regions based latitudes and longitudes. For example, the zone covering the equator within $\pm 9^{o}$ latitudes are divided into 18 regions. For 18° zones closer to north/south poles, they are divided to less regions to keep them roughly same area. The north pole centers around a single circular region within 9° from the pole. Similarly for the south pole. Etc. Grid points on higher elevation levels are grouped similarly, such that a single grid region looks like a column sticking out from sea-level and reaching up to upper atmosphere.

Note that the number of grids in each region differs substantially, even though all regions have similar surface area, because, for example, two points 2.5° longitudes apart are very close to each other when they are near the north/south poles. For this reason, grids along longitudinal direction are decimated when they are above 45°. As zones locate closer to poles, the rate of decimation increases, so that un-decimated grids in each regions become roughly same. The total number of grids is decimated to 8792. The matrix-vector multiplication will only use these un-decimated grids, and values at the decimated grids are simply interpolated from neighboring undecimated grids. This reduces the total computational efforts by 1- 8792/13104=31 %. The implementation keeps no grids decimation as a option so that one can check the consistency of the final solution.

How to distribute the grid regions on the processors? Since observations are already distributed, we distribute grid regions according to the sparsity pattern of the $P^{f}H^{T}$ matrix. On each processor, we loop through all 128 grid regions; if a grid region correlates to (i.e., being within 6000km distance from) at least one of the observation regions on the processor, this grid region maintains a copy on this processor. Since a given grid region correlates to many observation regions on many different processors, this grid region maintains many copies on different processors. In fact, on average, a grid region maintains copies on 1/3 of all the processors.

The matrix-vector multiplication proceeds on each processor by going through all correlated pairs between grid-regions and observation-regions. For each such pair, the matrix block is first calculated. For upper-air component, the matrix blocks between the observation region and the grid region at all pressure levels are calculated at once. (In this way, we can make use of the specific form of correlations to reduces the amount of required computation significantly at a later stage.) Then the matrix block and subvector multiplication is done and the result is appropriately accumulated using a single BLAS routine. These calculations are carried out independently and simultaneously on all processors.

Next we sum up appropriate increment sub-vectors on different processors to form the final results[3]. Since the grids regions on each processor differ, the order of increment sub-vectors are different too. So we reshuffle the sub-vectors on each processor into a universal vector which has identical order on all processors, of which the components not present on a processor are set to zero. Afterwards, a global sum over the universal vector on all processors are performed. This final results is written to a binary file on the disk. In the case where the grids been decimated, the length of the increment vector for each component at each level is restored to the original 13104, with decimated grids values being simple interpolations between nearest un-decimated grids along longitudinal direction.

5.0 Performance

Both parts of the parallel PSAS package is completed, and their accuracies are verified on smaller problems where the sequential results can be readily obtained. We carried out test runs on various problems on Intel Paragon with increasing problem sizes on increasing number of processors. Table 1 summarizes the CPU time for an 80,000 observation problem to all 14 pressure levels. The first half of the parallel package includes reading data from disk to all 512 processors, partitioning observations into 512 regions, generating load-balanced distribution lists for the correlation matrix, assembling the matrix blocks, and solving the correlation equation using a preconditioned CG solver. The second half includes generating grid regions, distributing/replicating folding matrix (first generating lists and then assemble the matrix blocks), carrying out the local matrix-vector multiplication, global summing of all increment vectors, restoring vector length/order and interpolating to obtain values on decimated grids. Note that 80,000 observation problem is practically the largest problem we can run on a C90. The real time, the wall-clock time, is much longer than the CPU time

TABLE 1. CPU time in seconds for a 79938 observations problem with 14 levels. * indicates estimates.

	Cray C90	Paragon 512-node
1st half	9120	75
2nd half	9000*	101
total	18000"	176

listed in the Table. Our parallel package reduces this 5hr C90 run to a 3min run. This 100+ -fold reduction in solution time is unprecedented in the area of high performance computing.

We also analyzed the performance of various parts of the package, The time-critical part of the package is the sparse linear equation solver, which achieves a sustained speed of 18.3 GFLOPS on a 512-node Paragon for an 85000 observation problem (see Figure 3.). This represents 36% of the theoretical total peak speed (512 * 100MFLOPS/node) of the 512-node Paragon. The solver achieves 77% peak speed on one node, 54% on 64 nodes, etc. These numbers indicate the high efficiency the package has achieved. The solver spends 27.5% of the time on communications for this problem on 5 12-node. Most of the communication time is spent on sending or receiving (average) 536 messages per processor with (average) 166 floating point numbers. These percentage numbers on peak total speed and on communication indicate the highly scalable" nature of the underlying algorithms.

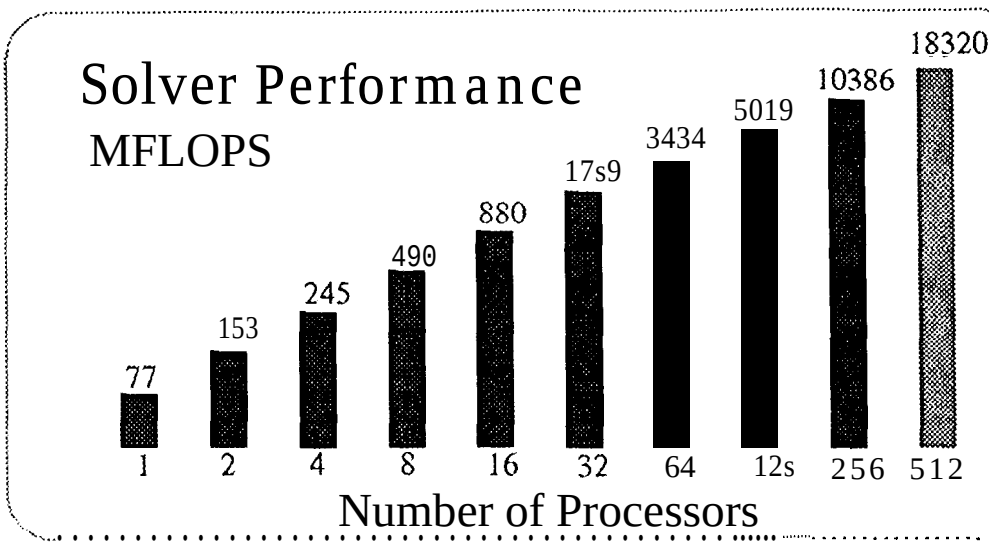

Fig.3 Performance of the equation solver.

6.0 Conclusions

We have designed and implemented a set of highly efficient and highly scalable algorithms for the PSAS data assimilation package, and achieved a 100+ -fold solution time reduction on the Intel Paragon parallel platform over Cray C90. This not only meets and exceeds the DAO time requirements, but also significantly enlarge the window of exploration in climate data assimilations.

Acknowledgment We thank Peter Lyster, Arlindo Da Silva, Jing Guo at DAO for making the Cray C90 codes available to us and helping us in understanding the problem. Access to the 512-node Paragon at Caltech is provided by the NASA HPCC Program. This work is carried out under a contract with NASA in HPCC ESS project.

References

[1] A. Da Silva, J. Pfaendtner, J. Guo, M. Sienkiewicz, and S.E. Cohn, Assessing the Effects of Data Selection with DAO'S Physical-space Statistical Analysis System, Proceedings of International Symposium on Assimilation of Observations, Tokyo, Japan, March, 1995.

[2] H.Ding and R. Ferraro, "Slices: A Scalable Concurrent Partitioned for Unstructured Finite Element Meshes" , Proceedings of SIAM 7th Conference for Parallel Processing, p.492, 1995.

[3] H.Ding and R. Ferraro, "A General Purpose Parallel Sparse Matrix Solvers Package", Proceedings of 9th International Parallel Processing Symposium, p.70, April 1995.